

Performance Modeling and Optimization of a High Energy Colliding Beam Simulation Code

Hongzhang Shan¹, Erich Strohmaier¹, Ji Qiang², David H. Bailey¹, and Kathy Yelick^{1,2}

Lawrence Berkeley National Laboratory¹, University of California at Berkeley²
{hshan, estrohmaier, jqiang, dhbailey, kayelick}@lbl.gov}

Abstract

An accurate modeling of the beam-beam interaction is essential to maximizing the luminosity in existing and future colliders. BeamBeam3D was the first parallel code that can be used to study this interaction fully self-consistently on high-performance computing platforms. Various all-to-all personalized communication (AAPC) algorithms dominate its communication patterns, for which we developed a sequence of performance models using a series of micro-benchmarks. We find that for SMP based systems the most important performance constraint is node-adapter contention, while for 3D-Torus topologies good performance models are not possible without considering link contention. The best average model prediction error is very low on SMP based systems with of 3% to 7%. On torus based systems errors of 29% are higher but optimized performance can again be predicted within 8% in some cases. These excellent results across five different systems indicate that this methodology for performance modeling can be applied to a large class of algorithms.¹

1 Introduction

Beam-beam interaction from electromagnetic force of charged particles is a dominant factor limiting luminosity in modern high-energy ring accelerators. In these colliders two beams of bunches of

electrical charged particles rotate against each other and collide at interaction regions. High luminosities are indispensable to achieving a high probability for particle-particle interaction during bunch collisions. Accurate modeling of the beam-beam interaction is essential to maximizing the luminosity in existing colliders and critical for building the next generation colliders such as Large Hadron Collider (LHC); any design errors will be enormously costly, or even fatal, for this “big science” project [1]. However, due to the extreme computational cost required to accurately and self-consistently model the beam-beam interaction as the beams circulate for millions of turns, previous studies are confined to using simplified models. Examples include the “weak-strong” model, in which only the “weak” beam is affected by the higher intensify “strong” beam [2], or soft Gaussian model [3], where one beam is assumed a priori to have a Gaussian shape. BeamBeam3D is the first parallel code that can be used to study this interaction fully self-consistently for both beams on high-performance computing platforms, including all the physical processes of long range off-centroid interactions, finite beam bunch length effects, and crossing angle collisions. The code has been used to study the beam-beam interactions at the world’s highest energy hadron accelerator currently used for experiments, Fermilab’s Tevatron, at SLAC’s Positron-Electron Project (PEP-II), at Brookhaven National Laboratory’s Relativistic Heavy Ion Collider (RHIC), and at Japan High Energy Accelerator Research Organization (KEKB).

BeamBeam3D is a challenging parallel code due to its high communication requirements and the need to simulate high number of beam-beam interactions iteratively. Simulating the beam-beam collision for millions of turns is extremely time-consuming and may currently take several weeks to finish. At present time a typical production run on the system Bassi at NERSC with 64 processors for 0.5 million turns takes 8 hours and represents

¹ Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. SC2006 November 2006, Tampa, Florida, USA

U.S. Government Work Not Protected By U.S. Copyright

about 10 seconds in physical time. For sufficient accuracy such a scenario should be simulated for 10 minutes to several hours of physical time, which requires between 100 and 1,000 times more compute cycles.

BeamBeam3D's dominant communication phases include a parallel grid reduction, during which each processor accumulates its local portion of a global, discretized charge density through a reduction of all its local grid elements from all other processors, a broadcast of electro-magnetic field to all other processors, and a forward-backward 2D FFT. All these phases represent different types of all-to-all personalized communication (AAPC). Furthermore, for some of these phases the communication volume per process stays constant in a strong scaling scenario, which results very fast in a communication bound execution with flat execution times at best. The problem offers multiple design, implementation, and optimization decisions such as particle or field domain decomposition based algorithms, different layouts for the processor grid, different strategies for localizing communication, and different algorithms for AAPC. This code is used on multiple, quite different parallel systems, and the tradeoffs between choices are not obvious and performance impacts are hard to judge, but multiple implementations of the code are also prohibitively expensive. For these reasons BeamBeam3D would benefit greatly from the development of a sufficiently accurate and flexible performance model for its different communication phases valid on a variety of modern parallel supercomputer architectures

BeamBeam3D is currently in use or being ported to an IBM SP Power3 (16 way SMP, Fat-tree), IBM SP Power5 (8 way SMP, Fat-tree), Opteron Infiniband cluster (2 way SMP, limited Fat-tree), Cray XT3 (single processor node, 3D Torus), and IBM BlueGene/L (single processor node, 3D Torus). This mix of systems represents a large variety of communication hierarchies with various levels of communication performance and networks with different contention points.

The rest of this paper is organized as follows: The computational methods of BeamBeam3D and the parallel implementations are described in Section 2. In Section 3 the five computational platforms used are described and the initial performance evaluation of Beambeam3D on them is presented. Section 4 presents the communication per-

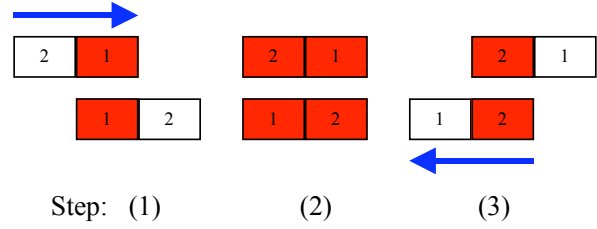


Figure 1: The colliding process for two bunches with two slices is simulated in three steps. Colliding slices are in red, beam1 is on top and beam2 is at the bottom.

formance model and the process of developing such a model. In section 5 we provide an overview of related work and our conclusions and future work are presented in Section 6.

2 BeamBeam3D

BeamBeam3D models the colliding process of two counter-rotating charged particle beams moving at close to the speed of light in a circular accelerator. Electrical charged particles move in a beam in separate bunches, which collide at specific points. Under the paraxial approximation, for the relativistic charged beam, the electric forces and the magnetic forces will cancel each other within the individual bunches [4]. However, for the colliding beams, which move in the opposite directions, the electric forces and the magnetic forces add up. The resulting beam-beam force produces a strongly nonlinear interaction that can significantly affect the motion of the charged particles. We use a multiple slice model to calculate the electromagnetic forces. In this model, each beam bunch is divided into a number of slices along the longitudinal direction in the moving frame of reference. Each slice contains nearly the same number of particles at different longitudinal locations z . The colliding process for two beams that have been divided into 2 slices is illustrated in Figure 1. During each step, only the red slices from opposite beams collide.

There are two important domains in Beambeam3D, particle domain and field domain. The particle domain is the configuration space containing the charged particles, and the field domain is the space where the electric field is generated by the charged particles. In the field domain decomposition, the whole computational domain is di-

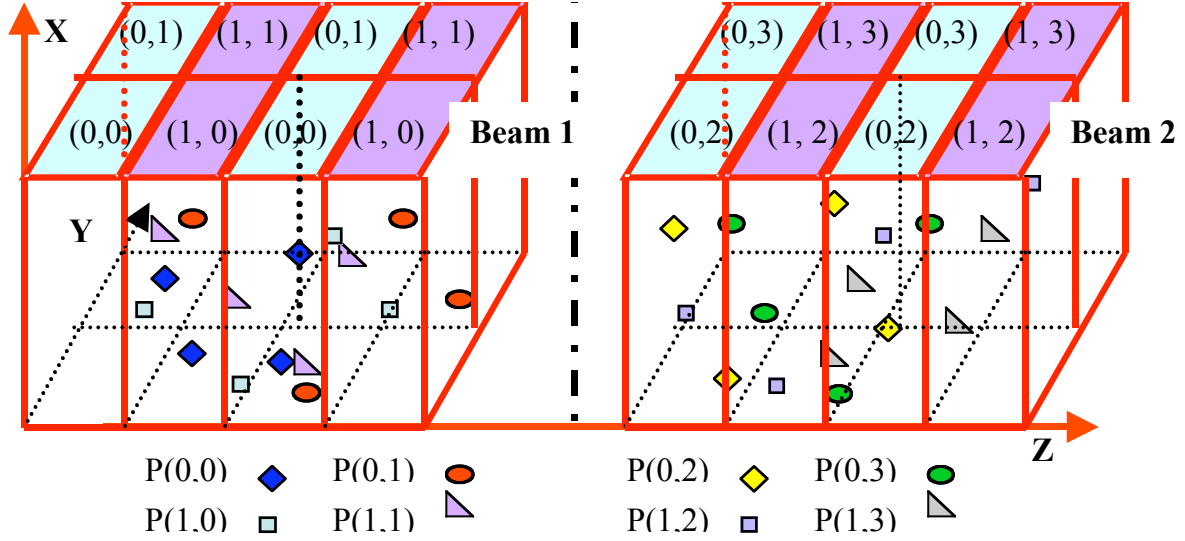


Figure 2: Illustration of the particle-field decomposition for eight processors. Each beam has four slices. The eight processors are divided into two groups and each group forms a 2x2 processor grid.

vided into a number of subdomains, and each subdomain together with the particles inside it is assigned to a processor. Since all particles are local to a processor, the Poisson equation, which has been used to compute the electric and magnetic forces for the field, is solved on the grid and the particles are advanced using the electromagnetic fields. However, in the next turn, the particles belonging to a subdomain may have moved to other subdomains. Therefore, the processors of different subdomains have to exchange particles with each other. If the particles move slowly, the domain decomposition approach may work well since most communication will be neighbor communication. However, due to the fact that in the accelerator, the lattice map outside the interaction point may cause significant particle movement, the effective communication pattern can end up as all-to-all communication and the data volume could be very large. A dynamic load balancer may potentially also be needed to balance the number of particles and the solution time of the Poisson equation among the processors.

BeamBeam3D adopts a novel particle-field decomposition approach to combine the advantages of both domain decomposition and particle decomposition, which has been demonstrated to deliver better performance than either particle decomposition or domain decomposition alone [6]. In this approach, each processor possesses the same

number of particles and the same number of computational grid points, i.e., a spatial subdomain of the same size. Figure 2 shows a schematic plot of the particle-field decomposition among eight processors. The total number of processors is divided into two groups, with each group responsible for one beam. We furthermore divide each beam longitudinally (z-direction) into a specified number of slices ($N_{\text{slice}}=4$ in Figure 2). The processors in each group are arranged logically into a two-dimensional array $P_z \times P_y$ to partition the computational domain, with each column (P_y) of the array

<i>Do K=1, number of turns</i>	<i># of turn to simulate</i>
<i>Do I=1, 2*Nslice-1</i>	<i># steps during the collision</i>
<i>Green2d()</i>	<i>2D FFT</i>
<i>Do I=1, Ncollide</i>	<i># colliding slices</i>
<i>Guardsum2d</i>	<i>Column reduction</i>
<i>Enddo</i>	
<i>Guardsum2drow</i>	<i>Row reduction</i>
<i>Do I=1, Ncollide/Prow</i>	<i># of colliding slices/Prow</i>
<i>Fieldsolver2d</i>	<i>2D FFT</i>
<i>Enddo</i>	
<i>Guardexch2drow</i>	<i>Row broadcasts</i>
<i>Do I=1, Ncollide</i>	<i># colliding slices</i>
<i>Guardexch2d</i>	<i>Column broadcasts</i>
<i>Enddo</i>	
<i>Enddo</i>	
<i>Enddo</i>	

Table 1: Dominant loop structure and communication steps in BeamBeam3D

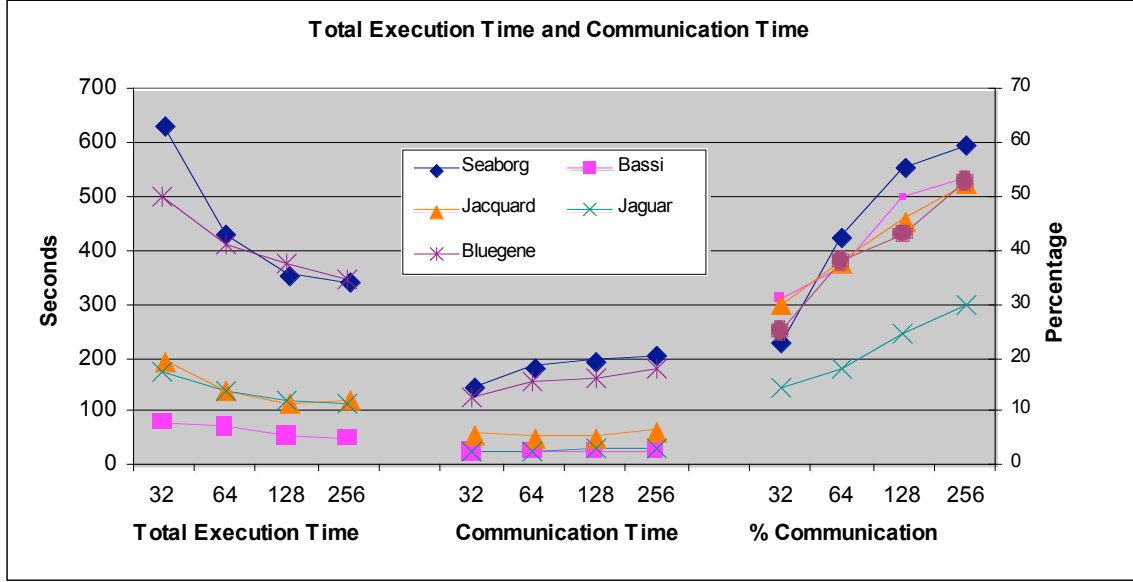


Figure 3: Total execution time, absolute, and relative communication time of BeamBeam3D. Times are given in seconds on the left and % of times on the right vertical axis.

containing a number of slices which are assigned to this column of processors cyclically along the longitudinal direction. This gives a good load balance of slices among different column processors. Within each column, the computational grid associated with each slice is decomposed uniformly among all the column processors. This allows us to parallelize the solution of the Poisson equation.

BeamBeam3D is implemented in Fortran90 using MPI. Table 1 summarizes the most important communication steps and the surrounding loop structures.

3 Performance Analysis

BeamBeam3D is used on several parallel computing systems. We selected five major systems for the development of the performance model and as target for our code optimization. Table 2 summarizes the main features of these five systems. This selection contains system with single processor and two, eight, and 16 way SMP nodes as well as interconnects with fat-tree or 3D torus topology using different technologies. The resulting global system architectures are hierarchical with substantially different locality structures. The full set represents a large variety of communication hierarchies with various levels of communication performance and networks with different contention points.

Name	System	Network	T.	SMP size	Proc. Peak [GF/s]	BW [MB/s]
Seaborg	SP Power3	Colony	FT	16	1.5	175
Bassi	SP Power5	Federation	FT	8	7.6	1112
Jacquard	Opteron	Infiniband	FT	2	4.4	360
Jaguar	XT3	SeaStar	3D	1	4.8	1084
BG/L	BlueGene/L	IBM custom	3D	1*	2.8	142

Table 2: Main architectural features of the system used in this study. Topology (T) is Fat-tree (FT) or 3D torus (3D); BW is bi-directional link bandwidth; (*On BG/L only one processor per node was used.)

Figure 3 shows the total execution time and communication time on all five systems across increasing concurrency levels for a fixed, typical problem size with 5 million particles, a grid size of $256^2 \times 8$ slices, and a process layout with 16 processes in column (y) direction (for up to 256 processors total). The limited scalability for this case is evident. Increasing the number of simulated particles would improve performance and scalability but does not represent the usage of this code. The chosen concurrency levels (32-256 processors) are ranging from typical levels up to maximum levels currently reasonably usable. While total execution time decreases (also asymptotically only little), communication time actually increases with increasing concurrency for all systems. This reflects the fact that the volume of communication for each processor is constant independent of the number of

Phase	Name	Pattern	Direction	Beam	Size [Byte]	# messages per turn
1:	Greenf2D	FFT Transpose	Column	Same	$(N_x/P_{col}+1)*$ $(N_y/P_{col})*16*2$	$(P_{col}-1) * (N_{slice}*2-1)$
2a:	Guardsum2D	All-to-All Reduce	Column	Same	$N_x*N_y/P_{col}*8$	$(P_{col}-1) * N_{slice} * N_{slice}$
2b:	Guardsum2Drow	All-to-All Reduce	Row	Same	$N_x*N_y/P_{col}*8*I$ $I = 1, N_{slice}/P_{row}$	$(P_{row}-1) * \text{MIN}(2*P_{row},$ $\text{CEILING}(N_{slice}/I, 1)*2-1)$
3:	Fieldsolver2D	FFT Transpose	Column	Same	$(N_x/P_{col}+1)*$ $(N_y/P_{col})*16$	$(P_{col}-1) * N_{slice} *$ $(N_{slice}+P_{row}-1)/P_{row} * 2$
4a:	Guardexch2Drow	All-to-All Broadcast	Row	Same	$N_x*N_y/P_{col}*8*I$ $I = 1, N_{slice}/P_{row}$	$(P_{row}-1) * \text{MIN}(2*P_{row},$ $\text{CEILING}(N_{slice}/I, 1)*2-1)$
4b:	Guardexch2D	All-to-All Broadcast	Column	Other	$N_x*N_y/P_{col}*8$	$P_{col} * N_{slice} * N_{slice}$

Table 3: Six most important communication phases and their main characteristics. N_x*N_y is the grid field size. $P_{row}*P_{col}$ is the processor grid as P_z*P_y mentioned earlier. N_{slice} is the number of slices a beam has been divided into ($P_{row} \leq N_{slice}$).

		Benchmark				
		PingPong	PingPing	Parallel PingPing		2D Grid
Used in Model		Model 1a	Model 1b	Model2	Model 3	Model 5
Range				Inter SMP	Intra SMP	256 proc
Seaborg	L [μ s]	50	29	92	18	100
	B [MB/s]	162	178	38	141	35
Bassi	L [μ s]	9.4	5.4	6.7	2.3	7.2
	B [MB/s]	825	1105	296	1660	264
Jacquard	L [μ s]	10.84	6.48	10.40	1.16	35.28
	B [MB/s]	303	360	185	714	140
Jaguar	L [μ s]	12.33	7.15	NA	NA	9.72
	B [MB/s]	544	1,084	NA	NA	346
BG/L	L [μ s]	5.4	4.1	NA	NA	3.2
	B [MB/s]	72.4	142	NA	NA	72

Table 4: Latency and bandwidth values of various micro-benchmarks used in the performance models. The parallel ping-ping benchmark uses a variable number of communication pairs depending on the systems SMP size. The 2D grid benchmark is run for the same concurrency levels as BeamBeam3D. Model 4 and 6 are missing as they are based on table lookups.

processors used. Such behavior imposes a severe limitation on scaling for any code. In Figure 3 we also show the percentage of communication time of the total execution time across concurrency levels. It confirms the initial conclusions and shows that on some systems for 256 processes communication time already amounts to up to 60% of total execution time.

4 Communication Performance Model

The first step in developing a performance model for the communication of a parallel code is to identify and to characterize the dominant communication phases. In Table 3 we show the six dominant communication phases along with their

major characteristics. It is important to notice that none of the communication pattern follows a simple nearest neighbor distribution. The transposes of the 2D FFTs (phases 1 and 3) take place within processors columns only, which in a typical case might contain 16 processors each. The parallel global grid reductions involve only processors within one beam and are organized in two phases communicating in column (phase 2a) or row (phase 2b) direction only. During the parallel global grid broadcast each processors has to send its part of the electromagnetic field to all members of the other beam. This is again organized in two phases, one in row direction within one beam (phase 4a) and the second one within column direction between beams (phase 4b). In phases 1, 2a,

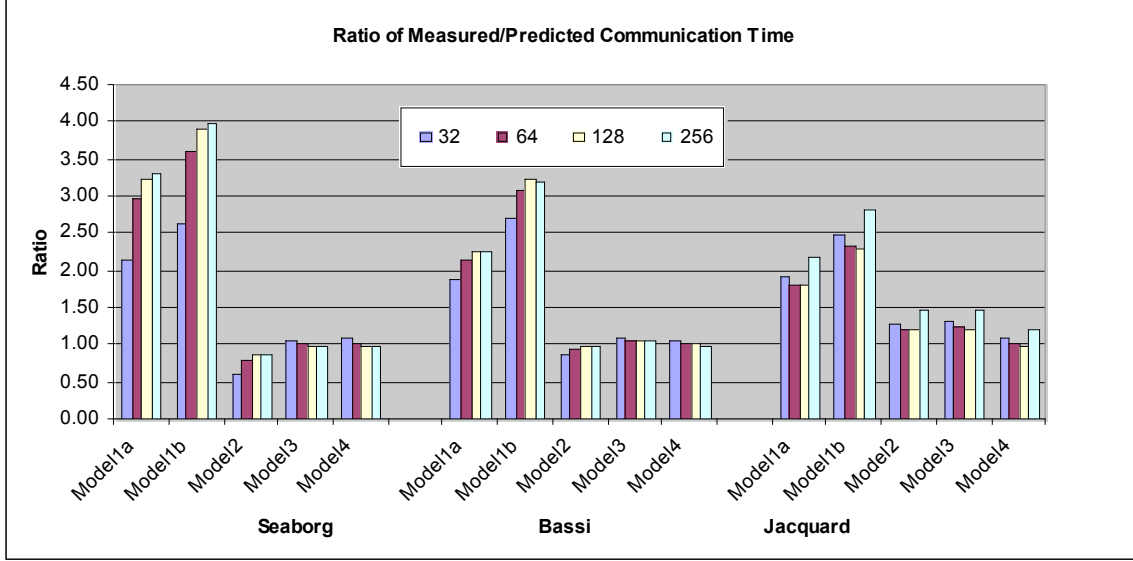


Figure 4: Ratio of measured to predicted total communication times of a sequence of five successively refined performance models for three SMP based systems.

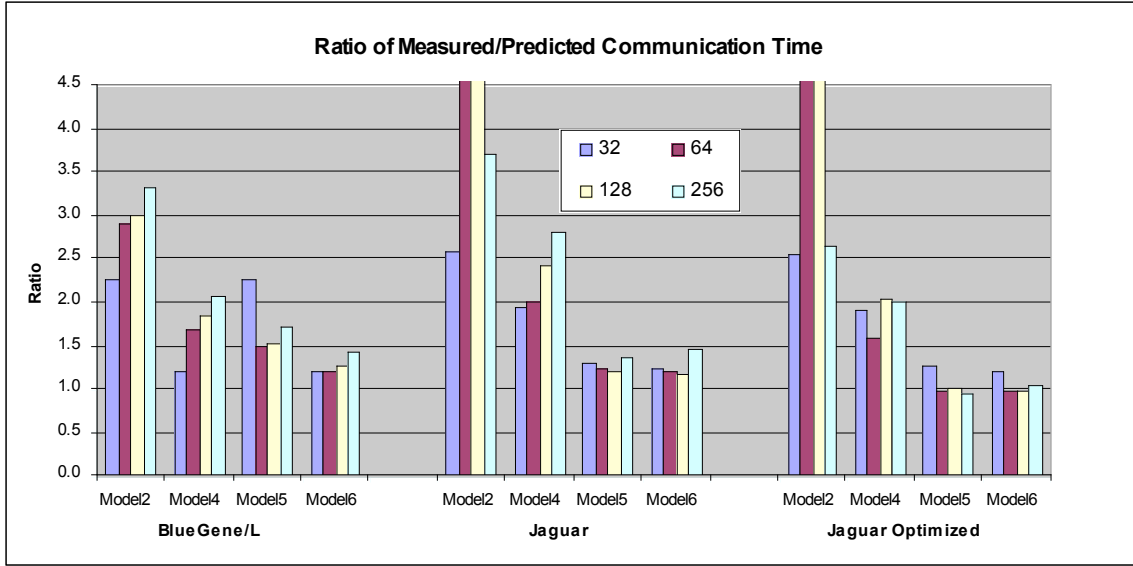


Figure 5: Ratio of measured to predicted total communication times of four performance models for the single-processor torus based systems. (*Jaguar models are shown for the original and for the optimized code.)

3, and 4b multiple messages might be exchanged between some pairs of processors, one for each slice involved in the communication, which explains message count larger than the number of involved processors. These messages could potentially be aggregated in single messages.

We chose a simple latency (L) and bandwidth (B) model for the time needed to exchange a single message of size s : $t = L + s/B$ and decided to measure the effective values of latency and band-

width using a synthetic micro-benchmark. We then estimate the communication time for each phase by summing up the individual message transfer times along its critical path, which is determined by the processor with the maximal volume of data and number of messages to send. We investigate a series of six performance models, which differ from each other by the type of micro-benchmark used to determine L and B , by separating different levels in the network hierarchy, and finally by replacing the

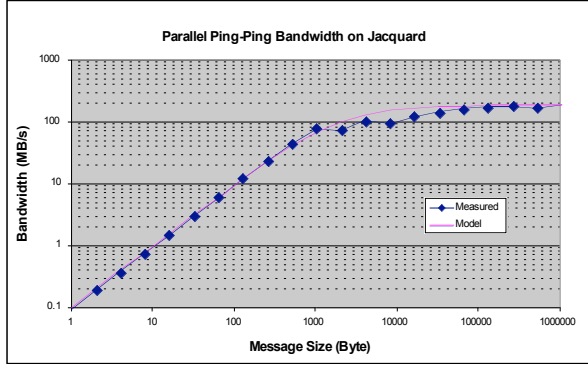


Figure 6: Message transfer times for the parallel ping-ping benchmark on Jacquard

linear timing model with table based lookups of the actual message transfer times.

For our first model we choose latency and bandwidth values measured with simple MPI message exchange benchmarks (ping-pong and ping-ping) between two processors on different nodes. Results are shown in Table 4 along with values from other benchmarks chosen for other performance models later on.

In Figure 4 we show the ratios of measured and predicted total communication times based on five of our models for the three SMP based systems in our study. The models based on point-to-point ping-pong or ping-ping benchmarks (model 1a, 1b) clearly under-predict the communication times by factors of 2 to 4.

For any model based on values measured with micro-benchmarks it is crucial that its parameters are chosen and measured in a fashion appropriate for the communication pattern in question. For all our communication phases (Table 3) all processors are communicating in pairs simultaneously. This implies that we cannot use latency and bandwidth numbers measured with simple point-to-point benchmarks, but that we have to use benchmarks, which replicate this pattern by using a sufficient number of communicating pairs of processors. For SMP based systems we typically have all processors of one SMP communicate with processors in a second SMP, which represent the largest load on the network between them possible and actually generated during the execution of BeamBeam3D. If we use parameters based on a parallel pair wise ping-ping benchmark for long-range communication (inter-SMP), the model prediction in Figure 4 (model 2) improves substantially, but the errors of the model are still noticeable. On the two IBM sys-

tem, which have 8 and 16-way SMP nodes, communication times are now over-predicted, while for the 2-way SMP based system Jacquard times are still under-predicted.

This indicates that the pattern of communication and the number of processors in the micro-benchmark also have to be chosen in such a way that the effects of potential communication bottlenecks are captured. Ignoring the hierarchical structure of SMP based system is one major source of errors in the model. This can be fixed by separating for each communication phase the messages within (intra) an SMP and between (inter) SMPs. This improves the error of the performance prediction (model 3) for the IBM system to less than 10%. The models for Jacquard still show errors of about 33%.

Further investigation reveals, that for the Opteron based Infiniband system Jacquard the approximation of message transfer times by a linear function in message sizes is not accurate enough to provide acceptable model predictions. Figure 6 shows that the actual transfer times for message sizes of interest between 4kB and 128kB are substantially different due to message protocol changes. If we replace our linear latency-bandwidth model of transfer times by a table lookup for actual achieved bandwidth values (model 4), prediction on Jacquard improves to within a few percent of measurement for concurrencies up to 128 processors and 21% for 256 processors. We believe that this increase in the error of the model prediction is due to contention in the network for large number of processors due to the fact that the second stage of the Infiniband switches on this system is not fully provisioned. We found further confirmation of this with the next micro benchmark described.

The current two-level hierarchy model of communication performance works well on the two IBM SP systems and the Opteron cluster, but the quality of prediction is still lacking for the other systems, Cray XT3 and IBM's BlueGene/L. Due to their single processor nodes, models 1b, 2, and 3 are identical. They under-predict times by roughly a factor of 2 to 10, which is improved by table loop-up (model 4) to a factor of 2 to 3 (Figure 5).

Both systems are different from the previous group of systems as they are using a 3D torus topology in their network instead of the more expen-

sive fat-tree networks. Networks with torus topology are more sensitive to network contention as long-range messages traverse multiple links and increase network load over-proportional. Capturing locality effects in these networks is difficult and requires more sophisticated models and/or benchmarks, which are sensitive to the average distance of processor in the typical communication patterns. As a first step we simulate these topology effects by arranging the communicating pairs of processors in the micro-benchmark in a 2D process grid similar to the process grid used in BeamBeam3D and by measuring performance along these different dimensions (model 5). As a consequence we now use latency and bandwidth numbers, which depend on the level of concurrency. This improves the model prediction for the XT3 drastically to within 20% to 35% (Figure 5), while BG/L predictions are still off by 50% to 130%. This model also produces predictions for the SMP based system of similar quality, as model 4 (not shown). The raw data from this benchmark shows a drop in effective performance on Jacquard for more than 128 processors, which confirms our previous analysis of our model accuracy on this system.

The linear latency, bandwidth model for our micro-benchmarks shows a similar deficiency on BG/L as on Jacquard, but with even larger deviation. As final model we therefore replace the linear latency and bandwidth model based on the 2D-grid benchmark with a table lookup based on concurrency level and message size (model 6). Prediction improves on BG/L to an average of 29% with a range of 20% to 45%. We are currently further investigating the reasons for these unusually large errors. Our current hypothesis is that effects of link-contention in BG/L are more difficult to model due to adaptive routing. Experiences during the optimization of AAPC for the Miranda code confirm that message injection rates for AAPC have to be controlled very carefully to achieve maximum performance [5]. Nevertheless, we believe that model accuracy even for BG/L has increased to a level to allow for some scalability predictions on this architecture.

Model 6 did not substantially improve prediction over model 5 for the Cray XT3 (27.9% over 28.4%). However, when we use the same model for the optimized versions of BeamBeam3D (next section), we find prediction errors of 7.6% for both

models. This is a strong indication, that our micro-benchmark does not capture some performance effects of the somewhat unusual initial process layout, but it works very well for the much more regular optimized layout.

5 Performance Optimization Strategies

The dominant communication phases of BeamBeam3D are the parallel grid reduction and the parallel grid broadcast. Both phases require each processor to send out a volume of data proportional to the global grid size. This represents a lower bound for the necessary communication volume per process, which cannot be exceeded and is the major restriction for optimizing communication performance in BeamBeam3D. Based on our experience during the development of the performance models we decided, that localization of communication was therefore the best available strategy to optimize communication performance. A second strategy is the aggregation of multiple messages between the same source and destination into single larger messages. This strategy could be used for most communication phases as they send multiple messages between column-processors, one for each individual slice involved. The maximum benefit of this strategy is the reduction of latency overhead. As our message sizes are not strongly latency bound, we decided to delay this optimization for later consideration.

For the FFT based phases with all-to-all communication along the process-column direction, this can be achieved by allocating processes along the column direction first. Initially processes were allocated randomly for production runs. The code could alternatively be executed by allocating groups of four processors along the row dimension first and we decided to use this alignment as basis for our study. On SMP based systems column orientation will result in the minimal number of SMPs being involved in single FFTs. This maximizes available bandwidth for all processes, as intra-SMP bandwidth is typically substantially higher than inter-SMP bandwidth. In Table 5 we show the resulting changes in message counts for phase 3 on SMPs of various sizes. Message sizes remain unchanged with this optimization.

Concurrency		SMP size					
		2		8		16	
		Intra SMP	Inter SMP	Intra SMP	Inter SMP	Intra SMP	Inter SMP
32	Base	128	1792	896	1024	1920	0
	Opt.	128	1792	896	1024	1920	0
64	Base	0	1080	216	864	504	576
	Opt.	72	1008	504	576	1080	0
128	Base	0	660	44	616	132	528
	Opt.	44	616	308	352	660	0
256	Base	0	450	0	450	30	420
	Opt.	30	420	210	240	450	0

Table 5: Message count of intra and inter SMP messages before and after localization of process allocation for phase 3.

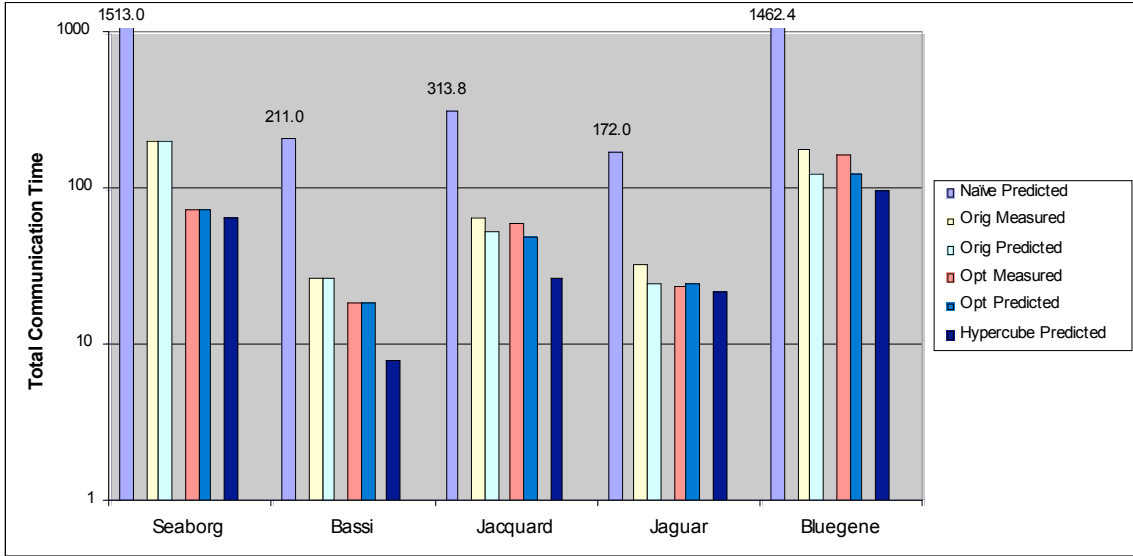


Figure 7: Measured and predicted communication times for various levels of optimizations for 256 processors

The parallel grid reduction and the parallel grid broadcast (phases 2 and 4) had already been optimized to some extent during the initial development of the code. For comparison purposes we used a performance model to predict the communication time for a straight forward implementation of these phases, in which each processor send p-1 or p messages to all other processors in the same or other beam. The initial code implementation breaks these two phases in two parts each by performing the reduction or broadcast operation in one dimension of the process grid first, reassembling the message and then performing the operation in the other dimension. Similar mesh algorithms have been studied previously for usage on mesh interconnect topologies [6], but the same type of algorithm can be used for parallel systems

with SMP nodes. However the initially chosen process layout proved to be sub-optimal and we were able to improve communication locality in such a way, that the phases with larger overall data-volume (phase 2a and 4b) again maximize data movement within SMP nodes. In addition during phase 4 at some point all data have to be exchanged between the processor groups assigned to the different beams. This was switched from phase 4b in the original code to phase 4a, as a substantially lower data volume is involved in the transfer in this phase, which again improves overall data locality of communication. Again, message sizes remain unchanged with all these optimizations.

Figure 7 shows the measured communication time before and after the described optimization together with the predicted times based on our best

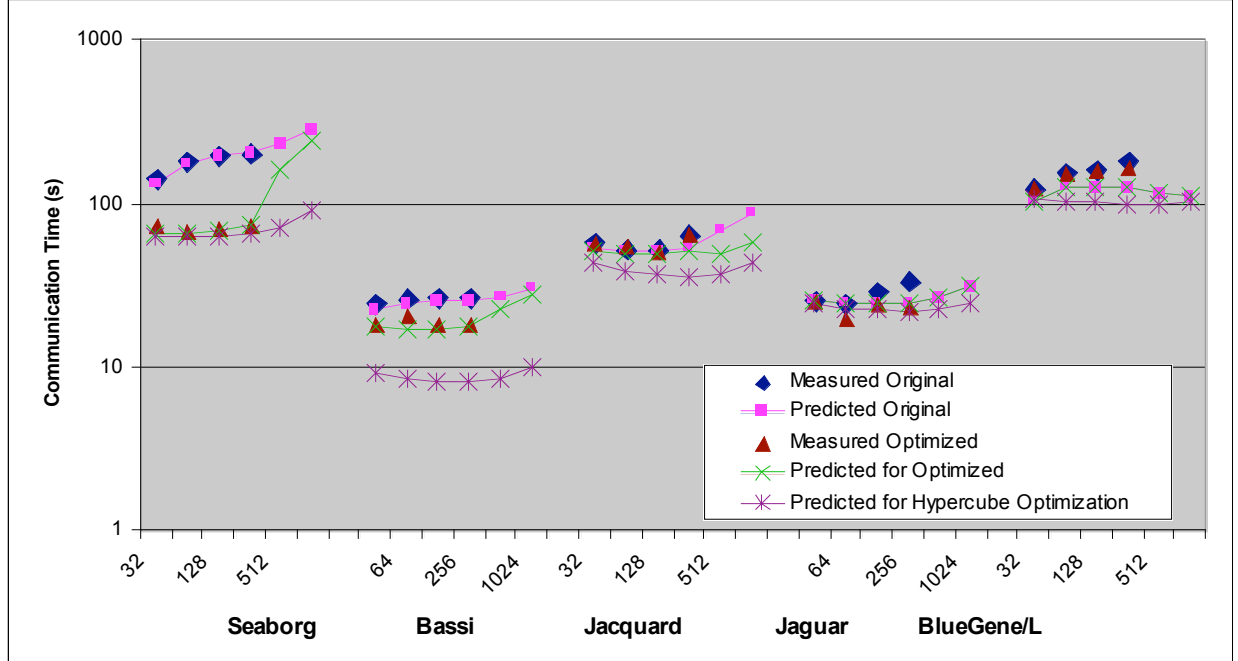


Figure 8: Measured and predicted scaling behavior of BeamBeam3D for up to 1024 processors.

performance model. The reduction in communication times for the SMP based system Bassi (up to 32%) and Seaborg (up to 65%) are substantial and well captured by our model. Improvements for Jacquard are only minor with 6% due to its small 2-way SMP size.

The impact of changing the process allocation policies on systems without SMP structure depends on the network topology and technology. For sufficiently connected network such as fat-trees there should in theory be no performance impact at all. For the torus networks localization of data transfers through process allocation which result in communicating large message over short distances and shorter message over longer distances might be beneficial, as network load and link contention is potentially reduced. However our current performance models do not incorporate such detailed network features and therefore predict essentially unchanged communication performance. On Jaguar performance actually improves up to 29%, while improvements for BG/L are only marginal with 8% on 256 processors.

An extension to using a two dimensional communication structure for improving the localization of message transfer is to generalize this idea to higher dimensions, which in the extreme leads to a hypercube structure. This results for the global grid reduction (phase 2) in sending largest

messages of half the global grid volume to nearest neighbors first and messages of half the previous size to twice as distant processors in later stages (linear order). For the global broadcast (phase 4) smallest messages of the local grid volume are first sent over the largest distance and subsequent messages of twice the size half as far. Due to the amount of work for such extensive recoding and time constraint we have not implemented this optimization yet, but decided to model its potential performance gains first. Predicted communication times for this optimization for 256 processors can also be seen in Figure 7.

To evaluate the overall impact of our present and potential optimizations, we show in Figure 8 the measured and predicted scaling behavior up to 1024 processors, based on the most accurate models for the currently considered restricted problem size. For this we assume that for more than 256 processors the number of slices and row-processors remains unchanged at 8 and the number of column processors increases from 16 to 64. Substantially higher concurrency levels are most likely not realistically usable for the current problem size due to the limitations of the current 2 dimensional parallelization strategy. Introducing a 3 dimensional parallelization strategy is currently not considered due to its complexity and uncertain pay-off.

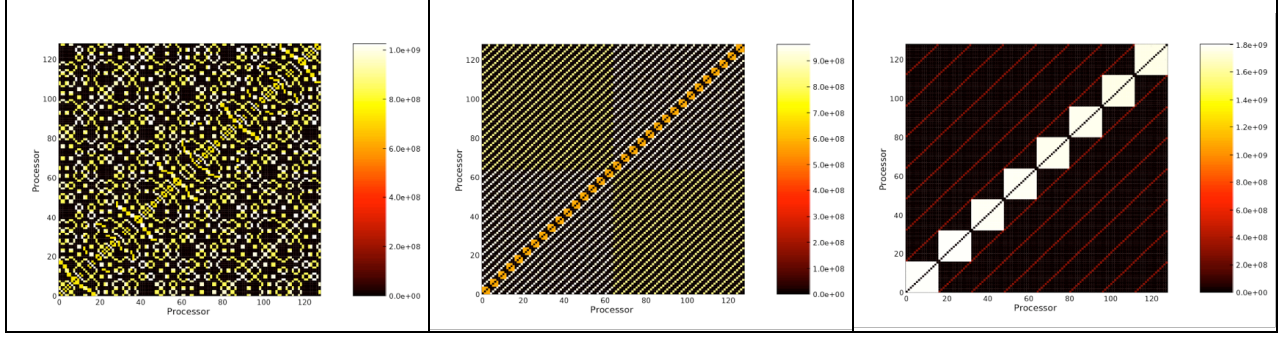


Figure 9: IPM heat-maps before and after optimization for 128 processors (2 beams * 4 rows * 16 columns). Each grid point is colored according to the volume of data communicated between processor i on the horizontal axis and j on the vertical axis. The left panel shows the communication pattern with random placement of processes, the middle column with the initial layout of blocks of 4 processes cyclic across rows, and the right panel after localizing both all communication phases. The improved localization of data transfers can clearly be seen as communication along the diagonal becomes more dominant.

Figure 8 clearly shows the improved scalability of the currently optimized code as well as the potential benefit of the hypercube based optimization. On Seaborg one process column of 16 processors fits completely within a single SMP and the benefit of the hypercube optimization therefore only appears for larger concurrency levels, where communication times might be reduced by 62% compared to the current optimized code. In contrast to this systems with smaller SMP nodes will benefit for any concurrency level as seen for Bassi (up to 65%) and even the 2-way SMP Jacquard (up to 28%). Predictions for Jaguar and BG/L improve only by up to 20%, which is not surprising, as our current models do not capture the details of the contention in the interconnects. Predicted improvements are completely caused by reduced message numbers and hence reduced latency overhead. Only substantially more complex model can provide better predictions on these systems.

The improved localization of communication can be visualized by using the performance tool IPM [7][8]. Based on profiles generated with IPM we visualize the communication volume between processors for both operation modes before and after optimization for 128 processors in Figure 9.

Overall the optimization strategies for BeamBeam3D were well guided by our performance models as well as the experience we gained in developing these models. A sufficiently accurate performance model can only be generated if the critical components of the system in question are well understood. Therefore the process of developing a

performance model naturally leads to a better understanding of the critical performance issues of a code on a specific system. In the case of BeamBeam3D the localization strategies for the FFTs and the parallel grid reduce/broadcast phases and the optimal choice of a beam-to-beam transfer point were a direct outcome of our performance models. In addition the models allow us to understand the potential benefits of hypercube based reduction/broadcast algorithm before considering its implementation.

6 Related Work

IMPACT [9] is a parallel 3D particle-in-cell code for modeling high intensity/high brightness beams in rf linear accelerators. The parallel implementation is based on a domain decomposition method. The particle movement across processors results in neighbor communication. The electromagnetic fields are calculated using a 3D FFT with a 2D logical processor array. Global communication is required at this stage. Therefore, the optimization strategy developed in this paper can also be applied to the IMPACT code.

The all-to-all personalized communication pattern (AAPC) has been studied in various contexts [6][10][11]. The algorithms used in the initial BeamBeam3D implementation for the global grid reduction and broadcast phases belong into the general class of SMP-aware shemes. However the global grid reduction algorithm has to our knowledge not been described in the literature.

Performance models are often based on architectural models such as LogP [12], which characterize the behavior of a system with a few parameters. AAPC have been studied in these frameworks [13]. Our models differentiate from this approach by ignoring the potential overlap of computation and communication, as we believe that the potential for optimizing the communication schedule itself should be exploited first. Unlike the original LogP, our models incorporate the effect of hierarchical interconnects, as we find them to be essential for modeling AAPC.

Other groups have developed performance models based on code inspection for several different scientific codes [14][15][16][17][18]. Naturally, as these models are code specific, they do not translate to other, different codes. Furthermore, often only the end-result of the model development is presented, while this study tries to document all necessary intermediate steps, to facilitate the dissemination of modeling techniques. Other emphasis, which set this study apart are the development of models for a larger variety of systems, as well as the integration of performance modeling into code optimization.

Other modeling approaches, which are based on (semi-) automatic code profiling, are also being developed [19]. Even as these approaches promise a significant reduction in model development time, they are still lacking the ability to predict performance beyond concurrency levels included in the profiling stage of the model development.

There are a large variety of synthetic micro-benchmarks available. We evaluate several of them [20][21][22], but due to the performance complexity of AAPC, none of them could be used to produce performance models with acceptable accuracy.

7 Conclusions and Future Work

In this study we developed performance models for a communication limited code simulating beam-beam interaction in high energy ring colliders (BeamBeam3D). The models were developed to predict communication across five different high performance computer systems with different architectures. We identified the most important communication phases, which all use all-to-all personalized communication patterns (AAPC). We started out with a simple ping-pong benchmark and

refined the benchmark iteratively to reflect the performance effects of the concurrent behavior of the main communication pattern, of different hierarchies in the system architectures, and of contention due to network topology.

On all systems performance can be modeled successfully with high accuracy. For SMP based systems the most important performance constraint is node-adapter contention. For 3D-torus based systems good performance models are not possible without considering the influence of link contention. Our current models do not incorporate a link-accurate contention model, which would be substantially more complicated. The average model prediction error for our best models is very low on the SMP based systems with 3% for Bassi, 4% for Seaborg, and 7% for Jacquard. On the torus based system they are somewhat higher with 28% for Jaguar and 29% for BG/L. However for Jaguar optimized performance is again predicted within 8%. Contention for AAPC on BG/L appears to be more difficult to model. These excellent results across five different systems indicate that this methodology for performance modeling can be applied to a large class of algorithms.

A 2D mesh based communication pattern for some of the AAPC shows very good result on SMP based systems. These algorithms are very similar to so called SMP-aware algorithms. Models predict that additional improvements on all systems are possible by using hypercube communication patterns instead.

Overall not only the performance but especially the parallel scalability of the code have been improved considerably. This allows us to increase concurrency levels for production runs without decreasing efficiency, which brings absolute code performance closer to targeted times.

8 Acknowledgments

This work was supported in part by the Department of Energy Office of Science through the SciDAC award High-End Computer System Performance: Science and Engineering. Computer time was provided by NERSC, Oak Ridge National Laboratory, and Argonne National Laboratory. We also like to thank Shoaib Kamil for his help with generating IPM plots.

References

- [1] J. Berkowitz, "Model Colliders", *DEIXIS 2004-2005 THE DOE Computational Science Graduate Fellowship ANNUAL*.
- [2] K. Hirata, H. Moshhammer, F. Ruggiero, "A symplectic beam-beam interaction with energy change", *Particle Accel.* 40 (1993) 205-228.
- [3] M.A. Furman, "Beam-beam simulations with the gaussian code TRS", *LBNL-42669*, CBP Note272, 1999.
- [4] J. Qiang, M.A. Furman, R.D. Ryne, "A parallel particle-in-cell model for beam-beam interaction in high energy ring colliders", *J. Comput. Phys.* 198 (2004) 278-294.
- [5] A. W. Cook, W. H. Cabot, P. L. Williams, B. J. Miller, B. R. de Supinski, R. K. Yates, M. L. Welcome: Tera-Scalable Algorithms for Variable-Density Elliptic Hydrodynamics with Spectral Accuracy. *SC 2005*: 60
- [6] V. Kumar, A. Grama, A. Gupta, and G. Karypis. Introduction to Parallel Computing. *The Benjamin/Cummings Publishing Company, Inc.*, 1994.
- [7] IPM: <http://ipm-hpc.sourceforge.net/>
- [8] L. Oliker, J. Borrill, J. Carter, D. Skinner, R. Biswas, "Integrated Performance Monitoring of a Cosmology Application on Leading HEC Platforms", *International Conference on Parallel Processing: ICPP 2005*, 2005.
- [9] J. Qiang, R. Ryne, S. Habib, V. Decyk, "An Object-Oriented Parallel Particle-In-Cell Code for Beam Dynamics Simulation in Linear Accelerators," *J. Comp. Phys.* vol. 163, 434, (2000).
- [10] D.R. Helman, D.A. Bader, and J. J'a'ja. Parallel Algorithms for Personalized Communication and Sorting With an Experimental Study. In *Proceedings of the Eighth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 211--220, Padua, Italy, June 1996.
- [11] S. Hinrichs, C. Kosak, D. O'Hallaron, T. Stricker, and R. Take, "An architecture for optimal all-to-all personalized communication," in *ACM Symposium on Parallel Algorithms and Architectures*, (Cape May, New Jersey), July 1994.
- [12] D. Culler, R. Karp, D. Patterson, A. Sahay, K. E. Schauser, E. Santos, R. Subramonian, and T. von Eicken, "LogP: Towards a realistic model of parallel computation", In *Proceedings 4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, May 1993.
- [13] R. M. Karp, A. Sahay, E. E. Santos, and K. E. Schauser, "Optimal Broadcast and Summation in the LogP model", In *Proc. Symposium on Parallel Algorithms and Architectures (SPAA)*, Velen, Germany, June 1993, 142--153
- [14] D. H. Bailey and A. S. Snaveley, "Performance Modeling: Understanding the Present and Predicting the Future," *Proceedings of Euro-Par 2005*, Lisbon, Portugal, Sep 2005;
- [15] J. Brehm, P. H. Worley, and M. Madhukar, "Performance Modeling for SPMD Message-Passing Programs", *Technical Report TM-13254*, Oak Ridge National Laboratory, June 1996
- [16] G. Mahinthakumar, M. Sayeed, J. Blondin, P. Worley, W. Hix, A. Mezzacappa, "Performance Evaluation and Modeling of a Parallel Astrophysics Application", in *Proceedings of the High Performance Computing Symposium 2004*, p. 27-33
- [17] A. Hoisie, O. Lubeck, and H. Wasserman, "Performance Analysis of Wavefront Algorithms on Very-Large Scale Distributed Systems". In *Lectures Notes in Control and Information Sciences*, 249 171, 1999.
- [18] M. M. Mathis, D. J. Kerbyson, A. Hoisie. "A Performance Model of non-Deterministic Particle Transport on Large-Scale Systems". In *Future Generation Computer Systems*, 2005. LA-UR 02-7313
- [19] A. Snaveley, L. Carrington, and N. Wolter, "Modeling Application Performance by Convolution Machine Signatures with Application Profiles," *Proc. IEEE Workshop on Workload Characterization*, 2001.
- [20] J. Dongarra, T. Hey, E. Strohmaier: "Selected Results from the ParkBench Benchmark". *Euro-Par 1996, Vol. II* 1996: 251-254
- [21] E. Strohmaier and H. Shan. "Apex-Map: A Global Data Access Benchmark to Analyze HPC Systems and Parallel Programming Paradigms", *Proceedings of SC05* (2005).
- [22] HPC Challenge Benchmark, <http://icl.cs.utk.edu/hpcc/>